Computational Power of P Systems with Small Size Insertion and Deletion Rules

Alexander Krassovitskiy
alexander.krassovitskiy@estudiants.urv.cat
Rovira i Virgili University,
Research Group on Mathematical Linguistics,
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain

Yurii Rogozhin
rogozhin@math.md
Rovira i Virgili University,
Research Group on Mathematical Linguistics,
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
Institute of Mathematics and Computer Science
Academy of Sciences of Moldova, Academiei, 5, MD-2028, Moldova

Sergey Verlan
verlan@univ-paris12.fr
Institute of Mathematics and Computer Science
Academy of Sciences of Moldova, Academiei, 5, MD-2028, Moldova
LACL, Département Informatique, Université Paris Est,
61 av. Général de Gaulle, 94010 Créteil, France

Recent investigations show insertion-deletion systems of small size that are not complete and cannot generate all recursively enumerable languages. However, if additional computational distribution mechanisms like P systems are added, then the computational completeness is achieved in some cases. In this article we take two insertion-deletion systems that are not computationally complete, consider them in the framework of P systems and show that the computational power is strictly increased by proving that any recursively enumerable language can be generated. At the end some open problems are presented.

1 Introduction

The operations of insertion and deletion are fundamental in formal language theory, and generative mechanisms based on them were considered (with linguistic motivation) for some time, see [9] and [2]. Related formal language investigations can be found in several places; we mention only [3], [5], [11], [13]. In the last years, the study of these operations has received a new motivation from molecular computing, see [1], [4], [15], [17], [10].

In general form, an insertion operation means adding a substring to a given string in a specified (left and right) context, while a deletion operation means removing a substring of a given string from a specified (left and right) context. A finite set of insertion-deletion rules, together with a set of axioms provide a language generating device (an InsDel system): starting from the set of initial strings and iterating insertion-deletion operations as defined by the given rules we get a language. The number of axioms, the length of the inserted or deleted strings, as well as the length of the contexts where these operations take place are natural descriptional complexity measures in this framework. As expected, insertion and deletion operations with context dependence are very powerful, leading to characterizations

T. Neary, D. Woods, A.K. Seda and N. Murphy (Eds.):
The Complexity of Simple Programs 2008.
EPTCS 1, 2009, pp. 108–117, doi:10.4204/EPTCS.1.10

of recursively enumerable languages. Most of the papers mentioned above contain such results, in many cases improving the complexity of insertion-deletion systems previously available in the literature.

Some combinations of parameters lead to systems which are not computationally complete [12], [6] or even decidable [18]. However, if these systems are combined with the distributed computing framework of P systems [14], then their computational power may strictly increase, see [7] where non-complete insertion-deletion systems of size (1,1,0;1,1,0) can generate any RE language, if considered in a P systems framework. In this paper we continue investigation of P systems with insertion-deletion and we show that P systems with insertion-deletion of size (2,0,0;1,1,0) and (1,1,0;2,0,0) are computationally complete, while pure insertion-deletion systems of the same size are not [8].

2 Prerequisites

All formal language notions and notations we use here are elementary and standard. The reader can consult any of the many monographs in this area – for instance, [16] – for the unexplained details.

We denote by $|w|$ the length of a word w and by $card(A)$ the cardinality of the set A.

An *InsDel system* is a construct $ID = (V,T,A,I,D)$, where V is an alphabet, $T \subseteq V$, A is a finite language over V, and I,D are finite sets of triples of the form (u,α,v), $\alpha \neq \varepsilon$, where u and v are strings over V and ε denotes the empty string. The elements of T are *terminal* symbols (in contrast, those of $V-T$ are called nonterminals), those of A are *axioms*, the triples in I are *insertion rules*, and those from D are *deletion rules*. An insertion rule $(u,\alpha,v) \in I$ indicates that the string α can be inserted in between u and v, while a deletion rule $(u,\alpha,v) \in D$ indicates that α can be removed from the context (u,v). As stated otherwise, $(u,\alpha,v) \in I$ corresponds to the rewriting rule $uv \to u\alpha v$, and $(u,\alpha,v) \in D$ corresponds to the rewriting rule $u\alpha v \to uv$. We denote by $\Longrightarrow_{ins}$ the relation defined by an insertion rule (formally, $x \Longrightarrow_{ins} y$ iff $x = x_1uvx_2, y = x_1u\alpha vx_2$, for some $(u,\alpha,v) \in I$ and $x_1,x_2 \in V^*$) and by $\Longrightarrow_{del}$ the relation defined by a deletion rule (formally, $x \Longrightarrow_{del} y$ iff $x = x_1u\alpha vx_2, y = x_1uvx_2$, for some $(u,\alpha,v) \in D$ and $x_1,x_2 \in V^*$). We refer by $\Longrightarrow$ to any of the relations $\Longrightarrow_{ins}, \Longrightarrow_{del}$, and denote by $\Longrightarrow^*$ the reflexive and transitive closure of $\Longrightarrow$ (as usual, $\Longrightarrow^+$ is its transitive closure).

The language generated by ID is defined by

$$L(ID) = \{w \in T^* \mid x \Longrightarrow^* w, x \in A\}.$$

The complexity of an InsDel system $ID = (V,T,A,I,D)$ is traditionally described by the vector $(n,m;p,q)$ called *weight*, where

$$\begin{aligned}
n &= \max\{|\alpha| \mid (u,\alpha,v) \in I\},\\
m &= \max\{|u| \mid (u,\alpha,v) \in I \text{ or } (v,\alpha,u) \in I\},\\
p &= \max\{|\alpha| \mid (u,\alpha,v) \in D\},\\
q &= \max\{|u| \mid (u,\alpha,v) \in D \text{ or } (v,\alpha,u) \in D\},
\end{aligned}$$

The *total weight* of ID is the sum $\gamma = m+n+p+q$.

However, it was shown in [18] that this complexity measure is not accurate and it cannot distinguish between universality and non-universality cases (there are families having the same total weight but not the same computational power). In the same article it was proposed to use the length of each context instead of the maximum. More exactly,

$$n = \max\{|\alpha| \mid (u,\alpha,v) \in I\},$$
$$m = \max\{|u| \mid (u,\alpha,v) \in I\},$$
$$m' = \max\{|v| \mid (u,\alpha,v) \in I\},$$
$$p = \max\{|\alpha| \mid (u,\alpha,v) \in D\},$$
$$q = \max\{|u| \mid (u,\alpha,v) \in D\},$$
$$q' = \max\{|v| \mid (u,\alpha,v) \in D\}.$$

Hence the complexity of an insertion-deletion system will be described by the vector $(n,m,m';p,q,q')$ that we call *size*. We also denote by $INS_n^{m,m'}DEL_p^{q,q'}$ corresponding families of insertion-deletion systems. Moreover, we define the total weight of the system as the sum of all numbers above: $\psi = n+m+m'+p+q+q'$. Since it is known from [18] that systems using a context-free insertion or deletion of one symbol are not powerful, we additionally require $n+m+m' \geq 2$ and $p+q+q' \geq 2$.

If some of the parameters n,m,m',p,q,q' is not specified, then we write instead the symbol $*$. In particular, $INS_*^{0,0}DEL_*^{0,0}$ denotes the family of languages generated by *context-free InsDel systems*. If one of numbers from the couples m, m' and/or q, q' is equal to zero (while the other is not), then we say that corresponding families have a one-sided context.

InsDel systems of a "sufficiently large" weight can characterize *RE*, the family of recursively enumerable languages.

An *insertion-deletion P system* is the following construct:

$$\Pi = (V,T,\mu,M_1,\dots,M_n,R_1,\dots,R_n),$$

where

- V is a finite alphabet,
- $T \subseteq V$ is the terminal alphabet,
- μ is the membrane (tree) structure of the system which has n membranes (nodes). This structure will be represented by a word containing correctly nested marked parentheses.
- M_i, for each $1 \leq i \leq n$ is a finite language associated to the membrane i.
- R_i, for each $1 \leq i \leq n$ is a set of insertion and deletion rules with target indicators associated to membrane i and which have the following form: $(u,x,v;tar)_a$, where (u,x,v) is an insertion rule, and $(u,x,v;tar)_e$, where (u,x,v) is an deletion rule, and tar, called the *target indicator*, is from the set $\{here, in, out\}$.

Any m-tuple $(N_1,\dots,N_n)$ of languages over V is called a configuration of Π. For two configurations $(N_1,\dots,N_n)$ and $(N_1',\dots,N_n')$ of Π we write $(N_1,\dots,N_n) \Longrightarrow (N_1',\dots,N_n')$ if we can pass from $(N_1,\dots,N_n)$ to $(N_1',\dots,N_m')$ by applying the insertion and deletion rules from each region of μ, in maximally parallel way, *i.e.*, in parallel to all possible strings from the corresponding regions, and following the target indications associated with the rules. We assume that every string represented in membrane has arbitrary many copies. Hence, by applying a rule to a string we get both arbitrary many copies of resulted string as well as old copies of the same string.

More specifically, if $w \in M_i$ and $r = (u,x,v;tar)_a \in R_i$, respectively $r = (u,x,v;tar)_e \in R_i$, such that $w \Longrightarrow_{ins}^r w'$, respectively $w \Longrightarrow_{del}^r w'$, then w' will go to the region indicated by tar. If $tar = here$, then the string remains in M_i, if $tar = out$, then the string is moved to the region immediately outside the

membrane i (maybe, in this way the string leaves the system), if $tar = in$, then the string is moved to the region immediately below.

A sequence of transitions between configurations of a given insertion-deletion P system Π, starting from the initial configuration $(M_1, \ldots, M_n)$, is called a computation with respect to Π. The result of a computation consists of all strings over T which are sent out of the system at any time during the computation. We denote by $L(\Pi)$ the language of all strings of this type. We say that $L(\Pi)$ is generated by Π.

We denote by $ELSP_k(insdel, (n, m, m'; p, q, q'))$(see, for example [14]) the family of languages $L(\Pi)$ generated by insertion-deletion P systems of degree at most $k, k \geq 1$ having the size $(n, m, m'; p, q, q')$.

3 Main results

Theorem 1. $ELSP_5(insdel, (1,1,0;2,0,0)) = RE$.

Proof. We prove the inclusion

$$ELSP_5(insdel, (1,1,0;2,0,0)) \supseteq RE$$

by simulating a type-0 grammar in Penttonen normal form by the means of insertion-deletion systems. The reverse inclusion

$$ELSP_5(insdel, (1,1,0;2,0,0)) \subseteq RE$$

is obvious as it follows from the Church thesis.

Let $G = (N, T, S, R)$ be a type-0 grammar in Penttonen normal form. This means that all production rules in R are of the form:

$$\begin{aligned} AB &\longrightarrow AC \text{ or} \\ A &\longrightarrow BC \text{ or} \\ A &\longrightarrow \alpha \end{aligned}$$

where A, B and C are from N and $\alpha \in T \cup N \cup \{\varepsilon\}$. Suppose that rules in R are ordered and $n = card(R)$. Now consider the following system.
$\Pi_1 = (V, T, [_1\ [_2\ [_3\ [_4\ [_5\ ]_5\ ]_4\ ]_3\ ]_2\ ]_1, \{SX\}, \emptyset, \emptyset, \emptyset, \emptyset, R_1, R_2, R_3, R_4, R_5)$.
It has a new nonterminal alphabet $V = N \cup T \cup \overline{P} \cup \{X\}, \overline{P} = \{P_i^j | i = 1, \ldots, n,\ j = 1, \ldots, 4\}$.

- For every production $i : AB \longrightarrow AC$ from R with $A, B, C \in N$ we add following rules to $R_1, \ldots, R_4$ correspondingly (we do not use membrane 5 in this case):

$$\begin{aligned} &(A, P_i^1, \varepsilon; in)_a \text{ to } R_1; \\ &(P_i^1, P_i^2, \varepsilon; in)_a \text{ and } (\varepsilon, P_i^1 P_i^3, \varepsilon; out)_e \text{ to } R_2; \\ &(\varepsilon, P_i^2 B, \varepsilon; in)_e \text{ and } (P_i^3, C, \varepsilon; out)_a \text{ to } R_3; \\ &(P_i^1, P_i^3, \varepsilon; out)_a \text{ to } R_4; \end{aligned}$$

- For every production $i : A \longrightarrow BC$ from R where $A,B,C \in N$ we add rules:

$$(A,P_i^1,\varepsilon;in)_a \text{ to } R_1;$$
$$(P_i^1,P_i^2,\varepsilon;in)_a \text{ and } (\varepsilon,P_i^2,\varepsilon;out)_e \text{ to } R_2;$$
$$(P_i^1,B,\varepsilon;in)_a \text{ and } (\varepsilon,P_i^3,\varepsilon;out)_e \text{ to } R_3;$$
$$(\varepsilon,AP_i^1,\varepsilon;in)_e \text{ and } (P_i^3,C,\varepsilon;out)_a \text{ to } R_4;$$
$$(P_i^2,P_i^3,\varepsilon;out)_a \text{ to } R_5.$$

- For every production $i : A \longrightarrow \alpha$ from R where $A \in N, \alpha \in T \cup N$ we add following rules to $R_1,\dots,R_4$ correspondingly (we do not use membrane 5 in this case):

$$(A,P_i^1,\varepsilon;in)_a \text{ to } R_1;$$
$$(P_i^1,\alpha,\varepsilon;in)_a \text{ and } (\varepsilon,P_i^2P_i^3,\varepsilon;out)_e \text{ to } R_2;$$
$$(P_i^1,P_i^2,\varepsilon;in)_a \text{ and } (P_i^2,P_i^3,\varepsilon;out)_a \text{ to } R_3;$$
$$(\varepsilon,AP_i^1,\varepsilon;out)_e \text{ to } R_4;$$

- For every production $i : A \longrightarrow \varepsilon$ from R with $A \in N$ we add rules $(\varepsilon,A,\varepsilon;here)_e$ to R_1.
- Finally, we add to R_1 rule $(\varepsilon,X,\varepsilon;out)_e$.

We claim that Π_1 generates the same language as G. In fact it is enough to proof that every step in derivation by grammar G can be simulated in Π_1.

Let us consider production $i : AB \longrightarrow AC \in R$.

The simulation of this rule is controlled by symbols P_i^1, P_i^2 and P_i^3. We assume that the sentential form in the skin membrane does not contain symbols from $\overline{P}$. Consider a string w_1ABw_2 in the skin region. We insert P_i^1 after symbol A : $w_1ABw_2 \Longrightarrow w_1AP_i^1Bw_2$ and send the obtained string to membrane 2. Here we insert P_i^2 after symbol P_i^1 : $w_1AP_i^1Bw_2 \Longrightarrow w_1AP_i^1P_i^2Bw_2$ and send the string to membrane 3. Next we delete substring P_i^2B : $w_1AP_i^1P_i^2Bw_2 \Longrightarrow w_1AP_i^1w_2$ and send the obtained string to membrane 4. Here we insert P_i^3 after P_i^1 : $w_1AP_i^1w_2 \Longrightarrow w_1AP_i^1P_i^3w_2$ and push the string to membrane 3. Now we insert symbol C after P_i^3 : $w_1AP_i^1P_i^3w_2 \Longrightarrow w_1AP_i^1P_i^3Cw_2$ pushing the string to membrane 2. Now we have two possibilities: to delete substring $P_i^1P_i^3$ and push the result w_1ACw_2 to the skin membrane (thus we simulate rule $i : AB \longrightarrow AC \in R$ correctly), or to insert symbol P_i^2 after P_i^1 and send string $w_1AP_i^1P_i^2P_i^3Cw_2$ to membrane 3, where symbol C will be inserted and the string comes back to membrane 2. So, we have a circle of computation in membrane 2 and 3. Notice, that between symbols P_i^1 and P_i^3 there is at least one symbol P_i^2, and therefore there is no possibility to apply rule $(\varepsilon,P_i^1P_i^3,\varepsilon;out)_e$ and to enter at the skin membrane. So, this branch of computation cannot influence the result and may be omitted in the consideration.

Let us consider production $i : A \longrightarrow BC$, where $A,B,C \in N$.

The simulation of this rule is controlled by symbols P_i^1, P_i^2 and P_i^3. We can also assume that the sentential form in the skin membrane does not contain symbols from $\overline{P}$. Consider a string w_1ABw_2 in the skin region. We insert P_i^1 after symbol A : $w_1ABw_2 \Longrightarrow w_1AP_i^1Bw_2$ and send the obtained string to membrane 2. Here we insert P_i^2 after symbol P_i^1 : $w_1AP_i^1Bw_2 \Longrightarrow w_1AP_i^1P_i^2Bw_2$ and send the string to membrane 3. Here we insert symbol B after P_i^1 : $w_1AP_i^1P_i^2w_2 \Longrightarrow w_1AP_i^1BP_i^2w_2$ and send the obtained

string to membrane 4. Here we delete substring AP_i^1 : $w_1AP_i^1BP_i^2w_2 \Longrightarrow w_1BP_i^2w_2$ and send the string to membrane 5. Now we insert symbol P_i^3 after P_i^2 : $w_1BP_i^2w_2 \Longrightarrow w_1BP_i^2P_i^3w_2$ and push the string to membrane 4. Here we insert symbol C after symbol P_i^3 : $w_1BP_i^2P_i^3w_2 \Longrightarrow w_1BP_i^2P_i^3Cw_2$ and push the string to membrane 3. Here we delete symbol P_i^3 and push the string to membrane 2: $w_1BP_i^2P_i^3Cw_2 \Longrightarrow w_1BP_i^2Cw_2$. At last we delete symbol P_i^2 and the result w_1BCw_2 enters at the skin region. So, we simulate rule $i: A \longrightarrow BC$ correctly.

Simulation of production $i: A \longrightarrow \alpha$, where $A \in N$ and $\alpha \in N \cup T$ is done in an analogous manner.

Every ε-production $i : A \longrightarrow \varepsilon, A \in N$ is simulated directly in the skin membrane by the corresponding rule $(\varepsilon, A, \varepsilon; here)_e$.

According to the definition of insertion-deletion P systems the result of a computation consists of all strings over T which are sent out of the system at any time during the computation. This is formally provided by the rule $(\varepsilon, X, \varepsilon; out)_e$ in the skin membrane. This rule uses conventional notation from [14]. Indeed, assume a sentential form wX appears in the skin membrane for some $w \in T^*$ (as we stared from the axiom SX). Then, applying the rule $(\varepsilon, X, \varepsilon; out)_e$ we assure that w is in $L(\Pi_1)$.

To claim the proof we observe that every correct sentential form has at most one symbol P_i^1, P_i^2 or P_i^3, $i = 1, \ldots, n$. And after insertion of P_i^1 in the skin membrane either all rules corresponding to i-th rule have to be applied (in the defined order) or the derivation is blocked. Hence, we have $L(G) = L(\Pi_1)$. $\square$

Theorem 2. $ELSP_5(insdel, (2,0,0;1,1,0)) = RE$.

Proof. We prove the inclusion

$$ELSP_5(insdel, (1,1,0;2,0,0)) \supseteq RE$$

by simulating a type-0 grammar in Penttonen normal form. The reverse inclusion

$$ELSP_5(insdel, (1,1,0;2,0,0)) \subseteq RE$$

follows from the Church thesis.

Let $G = (N, T, S, R)$ be a type-0 grammar in Penttonen normal form with production rules R are of type:

$$AB \longrightarrow AC \text{ or}$$
$$A \longrightarrow BC \text{ or}$$
$$A \longrightarrow \alpha$$

where A, B, C and D are from N and $\alpha \in T \cup N \cup \{\varepsilon\}$. Suppose that rules in R are ordered and $n = card(R)$. Now consider the following system.
$\Pi_2 = (V, T, [_1\ [_2\ [_3\ [_4\ [_5\ ]_5\ ]_4\ ]_3\ ]_2\ ]_1, \{SX\}, \emptyset, \emptyset, \emptyset, \emptyset, R_1, R_2, R_3, R_4, R_5)$.
It has a new nonterminal alphabet $V = N \cup T \cup \overline{P} \cup \{X\}$, $\overline{P} = \{P_i^j | i = 1, \ldots, n,\ j = 1, \ldots, 5\}$.

- For every production $i : AB \longrightarrow AC$ from R with $A, B, C \in N$ we add following rules to $R_1, \ldots, R_4$ correspondingly:

$(\varepsilon, P_i^1P_i^2, \varepsilon; in)_a$ to R_1;
$(P_i^2, B, \varepsilon; in)_e$ and $(A, P_i^3, \varepsilon; out)_e$ to R_2;
$(\varepsilon, P_i^3C, \varepsilon; in)_a$ and $(A, P_i^2, \varepsilon; out)_e$ to R_3;
$(A, P_i^1, \varepsilon; out)_e$;

- For every production $i : A \longrightarrow BC$ from R with $A,B,C \in N$ we add following rules to $R_1,\dots,R_5$ correspondingly:

$$(\varepsilon, P_i^1P_i^2, \varepsilon; in)_a \text{ to } R_1;$$
$$(P_i^2, A, \varepsilon; in)_e \text{ and } (\varepsilon, P_i^3, \varepsilon; out)_e \text{ to } R_2;$$
$$(\varepsilon, BP_i^3, \varepsilon; in)_a \text{ and } (P_i^3, P_i^2, \varepsilon; out)_e \text{ to } R_3;$$
$$(P_i^3, P_i^1, \varepsilon; in)_e \text{ and } (P_i^2, P_i^4, \varepsilon; out)_e \text{ to } R_4;$$
$$(\varepsilon, P_i^4C, \varepsilon; out)_a \text{ to } R_5;$$

- For every production $i : A \longrightarrow \alpha$ from R with $A \in N$, $\alpha \in N \cup T$ we add following rules to $R_1,\dots,R_4$:

$$(\varepsilon, \alpha P_i^3, \varepsilon; in)_a \text{ to } R_1;$$
$$(P_i^3, A, \varepsilon; in)_e \text{ and } (\alpha, P_i^2, \varepsilon; out)_e \text{ to } R_2;$$
$$(\varepsilon, P_i^1P_i^2, \varepsilon; in)_a \text{ and } (\alpha, P_i^1, \varepsilon; out)_e \text{ to } R_3;$$
$$(\alpha, P_i^3, \varepsilon; out)_e;$$

- For every production $i : A \longrightarrow \varepsilon$ from R with $A \in N$ we add the following rule to R_1: $(\varepsilon, A, \varepsilon; here)_e$.

- Finally, we add to R_1 the rule $(\varepsilon, X, \varepsilon; out)_e$.

Now we claim that Π_2 generates the same language as G. We show that every step in derivation by grammar G can be simulated in Π_2.

Let us consider production $i : AB \longrightarrow AC \in R$.

The simulation of this rule is controlled by symbols P_i^1, P_i^2 and P_i^3. As in the previous theorem, we assume that sentential form in the first membrane does not contain symbols from $\overline{P}$. Insertion of two symbols $P_i^1P_i^2$ sends the sentential form to the second membrane. As at this moment there are no symbols P_i^3 the only possible rule to be applied is $(P_i^2, B, \varepsilon; in)_e$. It assumes the presence of B on the right of P_i^2. This rule sends the sentential form to the third membrane. At this moment we can only apply the insertion $(\varepsilon, P_i^3C, \varepsilon; in)_a$ which sends the form to the forth membrane (hence $(A, P_i^2, \varepsilon; out)_e$ requires symbol A on the right from P_i^2). In the forth membrane we can apply the deletion rule $(A, P_i^1, \varepsilon; out)_e$ only if the first insertion $P_i^1P_i^2$ was done between A and B. Now we are pushed back to the third membrane. Here we have two options. The first option is to repeat the insertion $(\varepsilon, P_i^3C, \varepsilon; in)_a$. The derivation will be blocked in the next step as there is no symbols P_i^1 anymore. The second option is to apply $(A, P_i^2, \varepsilon; out)_e$. This is always possible since symbol P_i^2 appears adjacently right from A. This sends the sentential form to the second membrane. At this moment the sentential form does not contain any symbols from $\overline{P}$ except for P_i^3. And we can apply the deletion rule $(A, P_i^3, \varepsilon; out)_e$ assuming P_i^3C is inserted adjacently right from $P_i^1P_i^2$.

Hence, the only possible derivation by using the rules above is the following:

$$w_1ABw_2 \Longrightarrow w_1AP_i^1P_i^2Bw_2 \Longrightarrow w_1AP_i^1P_i^2w_2 \Longrightarrow$$
$$w_1AP_i^1P_i^2P_i^3Cw_2 \Longrightarrow w_1AP_i^2P_i^3Cw_2 \Longrightarrow$$
$$w_1AP_i^3Cw_2 \Longrightarrow w_1ACw_2.$$

One can see that this derivation correctly simulates the rule $i : AB \longrightarrow AC$.

Now we consider a context-free rule $i : A \longrightarrow BC$, where $A, B, C \in N$.

The simulation of the rule is controlled by symbols P_i^1, P_i^2, P_i^3 and P_i^4. The rule $(\varepsilon, P_i^1 P_i^2, \varepsilon; in)_a$ inserts $P_i^1 P_i^2$ and sends the sentential form to the second membrane. In the second membrane deletion rule $(P_i^2, A, \varepsilon; in)_e$ is applicable if $P_i^1 P_i^2$ is inserted adjacently left from A. It sends the form to the third membrane. Here, only insertion rule $(\varepsilon, BP_i^3, \varepsilon; in)_a$ is applicable as at this moment there are no symbols P_i^3 yet. It sends the form to the forth membrane. Here we can only delete P_i^1 as the rule $(P_i^2, P_i^4, \varepsilon; out)_e$ cannot be applied. In the fifth membrane we insert $P_i^4 C$ and the sentential form is pushed back to the forth membrane. At this step we can only remove P_i^4 and send the string to membrane 3. Now we have two possibilities: either insertion rule $(\varepsilon, BP_i^3, \varepsilon; in)_a$ or deletion rule $(P_i^3, P_i^2, \varepsilon; out)_e$ can be applied. In the first case the derivation will be blocked in membrane 4, as no rules may be applied to the string. In the second case symbol P_i^2 will be deleted and the string enters at membrane 2. Here symbol P_i^3 will be deleted and the result $w_1 BC w_2$ appears at the skin membrane.

Hence, the only possible derivation by using these rules is the following:

$$
\begin{aligned}
& w_1 A w_2 \Longrightarrow w_1 P_i^1 P_i^2 A w_2 \Longrightarrow w_1 P_i^1 P_i^2 w_2 \Longrightarrow \\
& w_1 B P_i^3 P_i^1 P_i^2 w_2 \Longrightarrow w_1 B P_i^3 P_i^2 w_2 \Longrightarrow \\
& w_1 B P_i^3 P_i^2 P_i^4 C w_2 \Longrightarrow w_1 B P_i^3 P_i^2 C w_2 \Longrightarrow \\
& w_1 B P_i^3 C w_2 \Longrightarrow w_1 B C w_2.
\end{aligned}
$$

So, we simulate rule $i : A \longrightarrow BC$ correctly.

Now, consider production $i : A \longrightarrow \alpha$ from R with $A \in N, \alpha \in N \cup T$. This case of replacement basically uses one insertion of αP_i^3 adjacently left from A, and two deletion rules $(P_i^3, A, \varepsilon; in)_e$ and $(\alpha, P_i^3, \varepsilon; out)_e$. But, hence, the total number of insertion-deletion rules for every production has to be even, we introduce one additional insertion $(\varepsilon, P_i^1 P_i^2, \varepsilon; in)_a$ and two deletion rules $(\alpha, P_i^1, \varepsilon; out)_e$, and $(\alpha, P_i^2, \varepsilon; out)_e$.

The derivation for this case has the following form:

$$
\begin{aligned}
& w_1 A w_2 \Longrightarrow w_1 \alpha P_i^3 A w_2 \Longrightarrow w_1 \alpha P_i^3 w_2 \Longrightarrow w_1 \alpha P_i^3 P_i^1 P_i^2 w_2 \Longrightarrow \\
& w_1 \alpha P_i^1 P_i^2 w_2 \Longrightarrow w_1 \alpha P_i^2 w_2 \Longrightarrow w_1 \alpha w_2
\end{aligned}
$$

So, we simulate rule $i : A \longrightarrow \alpha$ correctly.

Every ε-production $i : A \longrightarrow \varepsilon$, $A \in N$ is simulated directly in the skin membrane by the corresponding rule $(\varepsilon, A, \varepsilon; here)_e$. Finally, the rule $(\varepsilon, X, \varepsilon; out)_e$ is applied to wX in the skin membrane, where $w \in T^*$ and X is from the axiom SX. Here, we use the same technique as in the previous theorem. This rule is needed in order to terminate derivation and sent the resulting string as an output of the system.

In order to finish the proof we observe that every correct sentential form preserves the following properties:

1. No symbol from $\overline{P}$ presents in the skin membrane.
2. If some symbol from $\overline{P}$ appears more than once in the sentential form than the derivation is blocked on this production.

As shown before insertion of $P_i^1 P_i^2$ or BP_i^3 for the corresponding $i-th$ rule in the skin membrane results to either all rules corresponding to i-th rule have to be applied (in the defined order) or the derivation is blocked. Hence, we have $L(G) = L(\Pi_2)$. $\square$

4 Conclusions

In this article we have investigated P systems based on small size insertion-deletion systems. We proved two universality results, namely that insertion-deletion P systems with 5 membranes of size (2,0,0;1,1,0) and (1,1,0;2,0,0) are computationally complete. At the same time, pure insertion-deletion systems of the same size are not computationally complete. We guess that their computational power is rather small, but its precise characterizations is an open question. Another interesting question is whether the number of membranes used in the proof of Theorems 2 and 1 is minimal.

Finally, we would like to mention an interesting decidable class of insertion-deletion systems: systems of size $(2,0,0;2,0,0)$. We think that P systems with rules from this class will still not be able to generate any recursively enumerable language.

Acknowledgments

The first author acknowledges the grant of Ramon y Cajal from University Rovira i Virgili 2005/08 and grant no. MTM 63422 from the Ministry of Science and Education of Spain. The second author acknowledges the support of European Commission, project MolCIP, MIF1-CT-2006-021666. The second and the third author acknowledge the Science and Technology Center in Ukraine, project 4032.

References

[1] M. Daley, L. Kari, G. Gloor, R. Siromoney, Circular contextual insertions/deletions with applications to biomolecular computation. In: *Proc. of 6th Int. Symp. on String Processing and Information Retrieval, SPIRE'99* (Cancun, Mexico, 1999), 47–54.

[2] B.S. Galiukschov, Semicontextual grammars, *Matematika Logica i Matematika Linguistika*, Tallin University, 1981 38–50 (in Russian).

[3] L. Kari, *On insertion and deletion in formal languages*, PhD Thesis, University of Turku, 1991.

[4] L. Kari, Gh. Păun, G. Thierrin, S. Yu, At the crossroads of DNA computing and formal languages: characterizing RE using insertion-deletion systems. In: *Proc. of 3rd DIMACS Workshop on DNA Based Computing*, Philadelphia, 1997, 318–333.

[5] L. Kari, G. Thierrin, Contextual insertion/deletion and computability, *Information and Computation*, **131**, 1 (1996), 47–61.

[6] A. Krassovitskiy, Yu. Rogozhin, S. Verlan, Further results on insertion-deletion systems with one-sided contexts, Pre-proceedings of the 2nd International Conference on Language and Automata. Theory and Application. LATA 2008, March 13-19, 2008. Techical Reports of Research Group on Mathematical Linguistics, No. 36/08, 2008, 347–358.

[7] A. Krassovitskiy, Yu. Rogozhin, S. Verlan, One-sided Insertion and Deletion: Traditional and P Systems Case, Proceedings of International Workshop on Computing with Biomolecules, August 27th, 2008, Wien, Austria, 53–64.

[8] A. Krassovitskiy, Yu. Rogozhin, S. Verlan, About computational completeness of one-sided insertion-deletion systems (submitted).

[9] S. Marcus, Contextual grammars, *Rev. Roum. Math. Pures Appl.*, **14** (1969), 1525–1534.

[10] M. Margenstern, Gh. Păun, Yu. Rogozhin, S. Verlan, Context-free insertion-deletion systems. *Theoretical Computer Science*, **330** (2005), 339–348.

[11] C. Martin-Vide, Gh. Păun, A. Salomaa, Characterizations of recursively enumerable languages by means of insertion grammars, *Theoretical Computer Science*, **205**, 1–2 (1998), 195–205.

[12] A.Matveevici, Yu.Rogozhin, S.Verlan, Insertion-Deletion Systems with One-Sided Contexts. *Lecture Notes in Computer Science*, Springer, vol. 4664 (2007) 205-217.

[13] Gh. Păun, *Marcus contextual grammars*. Kluwer, Dordrecht, 1997.

[14] Gh. Păun, *Membrane Computing. An Introduction* (Springer–Verlag, Berlin, 2002), 163, 226–230.

[15] Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*. Springer–Verlag, Berlin, 1998.

[16] G. Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages*. Springer–Verlag, Berlin, 1997.

[17] A. Takahara, T. Yokomori, On the computational power of insertion-deletion systems. In: *Proc. of 8th International Workshop on DNA-Based Computers, DNA8* (Sapporo, Japan, June 10–13, 2002), *Revised Papers, LNCS*, **2568** (2003), 269–280.

[18] S. Verlan, On Minimal Context-Free Insertion-Deletion Systems. *Journal of Automata, Languages and Combinatorics* 12 (2007) 1/2, 317-328.